Esercitazione 4

29 novembre 2017

Termine per la consegna dei lavori: **martedì 5 dicembre** ore **23.55**.

Istruzioni

Ogni esercizio dovrà esser salvato in un file con estensione `.py` e consegnato tramite Moodle su sito `https://elearning.unipd.it/math`, sezione "**Matematica: Laurea Triennale**" corso di "**Laboratorio di Programmazione**" cliccando su "**Esecitazione4**".
È obbligatorio che all'interno di ogni file sia riportato il vostro nome, cognome e numero di matricola (potete riportarli all'interno di una riga commento all'inizio del file, es: `#Mario Rossi 1234567`).
Controllate che l'esecuzione del comando:

```
python <nome_file>.py
```

per ognuno degli esercizi produca l'output desiderato.

ATTENZIONE!

L'unico modulo importabile ammesso in questa esercitazione è il modulo `random`. Esercizi risolti utilizzando altri moduli importati riceveranno il punteggio minimo. Python contiene delle **built-in functions**, che potete utilizzare e la cui lista si può trovare a questo indirizzo: `https://docs.python.org/2/library/functions.html`.

Esercizio 1: Anti-Primi

Un naturale $n \geq 1$ è detto *anti-primo* (o "fortemente composto") se il numero dei suoi divisori è maggiore del numero di divisori di ogni numero più piccolo (≥ 1).
Esempio: i divisori del 6 sono 1,2,3 e 6 ovvero sono 4 che è maggiore del numero di divisori di tutti i numeri dall'1 al 5.

Scrivere un programma che cerchi, e stampi a video, i primi 10 numeri anti-primi.

Esercizio 2: descrittori di lista

Scrivere un programma che crei la lisa L, avente ad ogni indice $0 \leq i \leq 99$, il numero dei divisori di $i+1$ se questi sono al più 5, -1 se i è multiplo di 3 ma non di 5, altrimenti $i+1$.

$$L[i] = \begin{cases} \text{il numero di divisori di } i+1 \text{ se sono al più 5, altrimenti} \\ -1 \text{ se il numero di divisori di } i+1 \text{ è multiplo di 3 ma non di 5} \\ \text{altrimenti } i+1 \end{cases}$$

Esercizio 3: alberi

Dato un intero $n > 0$, scrivere un programma in python che stampi un albero a forma di abete su $n+1$ livelli utilizzando ripetizioni del carattere $'*'$. Ogni livello conterrà due $'*'$ in più rispetto il precedente, ad eccezione del primo che ne conterrà uno solo, e dell'ultimo che conterrà il fusto, rappresentato da un carattere $'*'$.
Ogni livello inoltre, dovrà esser centrato opportunamente in maniera tale da formare un albero bilanciato e simmetrico. Risolvere l'esercizio utilizzando **solo** il ciclo while. Segue un esempio con $n = 3$:

```
  *
 ***
*****
  *
```

Esercizio 4: tic-tac-toe

Scrivere un programma che simuli il gioco del *tic-tac-toe* (tris) tra CPU1 e CPU2. Partendo da un giocatore scelto casualmente, ogni turno vedrà i due alternarsi per posizionare una pedina su una cella di una griglia 3×3. La cella scelta non dev'esser già occupata da un'altra pedina. Il gioco termina quando:

- uno dei due giocatori vince effettuando un *tris*, ovvero un allineamento in orizzontale, verticale o diagonale, di 3 pedine da lui posizionate;
- non è possibile effettuare ulteriori mosse in quanto ogni cella della griglia di gioco è già occupata. In tal caso la partita finisce in parità.

Il programma terminerà quando uno dei due giocatori vincerà 3 partite consecutive, stampando in output il vincitore, il numero di partite effettuate e la percentuale di vittoria di entrambi i giocatori. Gestire opportunamente le strutture dati da utilizzare.